Éditeur et analyseur de réseaux de Petri (avec extension Réseaux de Petri colorés)

Ingénieur designer, Théorie des graphes

14 septembre 2025

1 Contexte et objectifs

Dans ce projet, il vous est demandé de concevoir et implémenter un *éditeur et analyseur de réseaux de Petri*, implémenté en **Python**, avec un rendu documenté en LaTeX. Le projet couvre :

- l'édition (graphique ou textuel) de réseaux de Petri classiques ;
- la génération et l'exploration de l'espace d'états (graphe d'états) ;
- la vérification de propriétés classiques (deadlock, vivacité, bornitude, invariants, etc.) ;
- une extension pour réseaux de Petri colorés (CPN) et leur gestion (types, expressions, couleurs) ;

2 Exigences fonctionnelles

2.1 Éditeur

- L'utilisateur peut créer, modifier et supprimer les éléments suivants : places, transitions, arcs, annotations (étiquettes).
- Chaque place doit pouvoir contenir un marquage initial (entier ou multiensembles pour CPN).
- Les arcs peuvent être orientés et posséder des inscriptions (poids, expressions pour CPN).

2.2 Analyse formelle

- Génération de l'espace d'états (graphe de reachability) à partir d'un marquage initial donné.
- Calcul et export du graphe d'états (format GraphML / DOT / PNG). Pour CPN, proposer une extension ou exporter vers un format JSON documenté.
- Vérification automatique des propriétés : *deadlock* (états sans transition activable), bornitude (boundedness), vivacité (liveness), conservation (place invariants), transition invariants.

3 Architecture logicielle proposée

3.1 Vue d'ensemble

Le logiciel sera organisé en couches :

- **UI** : éditeur, vues de l'espace d'états et panneau d'exploitation de l'espace d'états.
- **Modèle** : classes ou modules représentant Place, Transition, Arc, Net, Marking, CPN types.
- **Moteur** : simulateur, explorateur d'états, vérificateur de propriétés.
- **Persistance** : modules d'import/export (PNML, JSON), gestion des projets.

3.2 Algorithmes importants

- Exploration BFS/DFS avec table de hachage des marquages (ex. dictionnaire Python avec tuple normalisé comme clé).
- Détection de bornitude : stockage des bornes rencontrées et détection d'un dépassement de seuil configuré.
- Et d'autres algorithmes que vous jugez nécessaires.

4 Spécification technique détaillée

5 Extension : Réseaux de Petri colorés (CPN)

5.1 Fonctionnalités CPN

- Définition de types de couleurs (entier, énumération, produit) et de variables.
- Expressions d'arc et gardes sur transitions.
- Fonction d'instanciation des couleurs pour le marquage initial.

6 Tests et validation

- Tests unitaires pour le modèle, le moteur et l'I/O (pytest).
- Tests d'intégration pour scénarios classiques (exemples de réseaux pédagogiques).
- Mesures de performance : temps d'exploration, consommation mémoire sur jeux d'essai.

7 Livrables

1. Code source complet (dépôt Git) avec README et guide d'installation.
2. Documentation utilisateur (fichier LaTeXcompilable + PDF généré).
3. Jeux d'exemples (nets PNML / JSON) et jeux de tests.

Références

[1] Tadao Murata. Petri Nets : Properties, Analysis and Applications. *Proceedings of the IEEE*, 77(4) :541–580, 1989.

[2] Wolfgang Reisig. *Understanding Petri Nets : Modeling Techniques, Analysis Methods, Case Studies*. Springer, 2013.

[3] Kurt Jensen. *Coloured Petri Nets : Basic Concepts, Analysis Methods and Practical Use. Volume 1–3*. Springer, 1997–2009.

[4] Kurt Jensen and Lars M. Kristensen. *Coloured Petri Nets : Modelling and Validation of Concurrent Systems*. Springer, 2015.

[5] ISO/IEC 15909-2. High-level Petri nets – Part 2 : Transfer format. International Organization for Standardization, 2011.